Soft Computing (2020) 24:981–995
https://doi.org/10.1007/s00500-019-03937-0

METHODOLOGIES AND APPLICATION

Multi-objective redundancy hardening with optimal task mapping for independent tasks on multi-cores

Bo Yuan[1] · Bin Li[2] · Huanhuan Chen[3] · Zhigang Zeng[4] · Xin Yao[1,5]

Published online: 27 March 2019

Abstract

The rate of transient faults has increased significantly as the technology scales up. The tolerance of transient faults has become an important issue in the system design. Dual modular redundancy (DMR) and triple modular redundancy (TMR) are two commonly used techniques that can achieve fault detection and masking through executing redundant tasks. As DMR and TMR have different time and cost overheads, we must carefully determine which one should be used for each task (i.e., task hardening) to achieve the optimal system design. Furthermore, for multi-core systems, the system-level design includes the allocation of cores for the tasks (i.e., task mapping) as well. This paper aims at task hardening and mapping simultaneously for independent tasks on multi-cores with heterogeneous performances, in order to minimize the maximum completion time of all tasks (i.e., makespan). We demonstrate that once task hardening is given, task mapping of independent tasks can be achieved by employing min–max-weight perfect matching with a polynomial time complexity. Besides, as there is a trade-off between cost and time performance, we propose a multi-objective memetic algorithm (MOMA)-based task hardening method to obtain a set of solutions with different numbers of cores (i.e., costs), so the designer can choose different solutions according to different requirements. The key idea of the MOMA is to incorporate problem-specific knowledge into the global search of evolutionary algorithms. Our experimental studies have demonstrated the effectiveness of the proposed method and have shown that by combining the results of MOMA and MOEA we can provide a designer with a highly accurate set of solutions within a reasonable amount of time.

Keywords Fault tolerance · Multi-cores · Task hardening · Task mapping · Multi-objective optimization · Memetic algorithms

Communicated by V. Loia.

✉ Xin Yao
xiny@sustc.edu.cn

Bo Yuan
yuanb@sustc.edu.cn

Bin Li
binli@ustc.edu.cn

Huanhuan Chen
hchen@ustc.edu.cn

Zhigang Zeng
zgzeng@hust.edu.cn

[1] Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China

[2] School of Information Science and Technology, University of Science and Technology of China, Hefei 230026, China

[3] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China

[4] School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China

[5] CERCIA, School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

1 Introduction

The rapid technology advances, such as transistor size scaling, high operation frequency, and low voltage supply, have led to multiple reliability threats to circuits and systems (Constantinescu 2003). For example, the rate of transient faults in circuits has dramatically increased in the past years. A transient fault is usually caused by high-energy particle that strikes to flip the state of bits in an unpredictable way, which can corrupt the correct application execution state (Baumann 2005). Since the rate of transient faults is much higher than that of permanent faults, any reliable system should employ an effective scheme to tolerate transient faults in the future (Henkel et al. 2013a).

Multi-cores have emerged to be a popular and powerful computing platform for many recent systems (Ebi et al. 2009; Henkel et al. 2013b; Jahn et al. 2011). As the technology keeps scaling down, more and more cores can be integrated into a single chip to meet the growing computing demand of modern applications. Such applications depend on high-performance computing (HPC), and a high reliability in the presence of transient faults is required as well. On the other hand, the individual cores may exhibit significant frequency variations (e.g., up to 30%) because of process variations (result from fabrication imprecision) (Bowman et al. 2002; Dighe et al. 2011). This situation is worsening with technology scaling because of the difficulty of precise fabrication with reduced dimensions at a nanoscale.

To achieve reliability against transient faults, three types of redundancies have conventionally been applied, i.e., time redundancy, space redundancy, and hybrid redundancy. Time redundancy (i.e., re-execution) is based on checkpointing and to execute the task again in case of a fault (Kandasamy et al. 2003; Nikolov and Larsson 2016; Salehi et al. 2016a); it is applicable only after a fault can be detected. The overhead of local fault detection is not free since it is hard to achieve perfect transient fault detection. Space redundancy (i.e., replication), based on executing redundant replicas of task independently on different cores, does not require any specific fault detection mechanism and uses result comparison (majority voting) for fault detection and masking (Koren and Krishna 2007; Pradhan 1996; Salehi et al. 2016b). To make a majority voting, the number of replicas should be an odd number so that triple modular redundancy (TMR) is commonly used (Chen et al. 2016; Lyons and Vanderkulk 1962). Although TMR is simple and predictable for tasks with deadlines, it requires more resources and energy than dual modular redundancy (DMR) (Dave and Jha 1999; Vadlamani et al. 2010). On the other hand, DMR is only capable of detecting fault since it is unable to decide which one is correct. Therefore, another way to implement fault masking is to combine DMR-based detection and checkpoint-based re-execution, which can be regarded as the hybrid redundancy (Kang et al. 2015; Pradhan and Vaidya 1994; Ziv and Bruck 1997). Both TMR and DMR are well suited for multi-core platforms as multi-cores can provide multiple processing units and low-overhead communication for comparing and voting.

Since different fault-tolerant techniques are usually characterized by different time and space overheads, there is an optimization trade-off in *task hardening*, i.e., determining one of the fault-tolerant techniques (e.g., DMR or TMR) for each task. This trade-off, together with the traditional optimization in *task mapping*, i.e., mapping each task to one of the cores, makes the design of fault-tolerant multi-cores very challenging (Das et al. 2014; Gan et al. 2012; Khosravi et al. 2014; Pop et al. 2009; Stralen and Pimentel 2012). It is notable that *replication*-based *task hardening* will introduce new tasks, i.e., replicas, into the system, and these replicas should be mapped as well. In this case, the design of fault-tolerant multi-cores can be regarded as a bi-level optimization problem due to its nested structure. Specifically, the upper-level optimization subproblem is *task hardening* and the lower-level optimization subproblem is *task (including all original tasks and new tasks) mapping*, thereby making the overall optimization computationally very intensive.

As both *task hardening* and *task mapping* are NP-hard in general cases (Bolchini et al. 2011), most of the previous methods are based on meta-heuristics, especially evolutionary algorithms (EAs). Through mapping a task to a set of cores, *replication*-based *task hardening* can be implicitly implemented, e.g., the number of mappings and the list of cores for each task are encoded as an integer vector in (Glaß et al. 2007; Huang et al. 2011; Huang et al. 2012; Reimann et al. 2008), or the mappings to all cores for each task are encoded as a 0–1 vector in Kang et al. (2014a, b). As *task hardening* is implicitly implemented in *task mapping*, the chromosome length of *task mapping* must be overestimated according to the maximum number of replicas that are probably introduced by *replication*-based hardening (Glaß et al. 2007; Huang et al. 2011; Huang et al. 2012; Reimann et al. 2008) or introduced by the total number of cores on the multi-core platforms (Kang et al. 2014a, b). The problem of overestimation is relaxed in the nested two-layer EA (Bolchini and Miele 2013; Bolchini et al. 2011), where the outer layer EA is used to explore *task hardening*, while the inner layer EA is used to explore *task mapping* on each extended task graph (Jhumka et al. 2005) independently. Nested two-layer structure is a commonly used framework for bi-level optimization problems, but the EA-based inner layer makes the whole optimization process very time-consuming. Tabu search (TS) seems to be an alternative way of exploring both *task hardening* and *task mapping* at the same time (Izosimov et al. 2009; Lifa et al. 2010). As an individual/trajectory-based metaheuristic, the encoding strategy of TS is simple and flexible. But the performance of TS highly depends on the initial solution which is generated greedily in the TS-based methods, and generally TS is easy to get trapped in local optima, especially for problems with complex landscape.

As stated above, it is very difficult to develop efficient method for joint *task hardening* and *task mapping* for general cases, because of the nested structure of the two NP-hard problems. This paper builds upon the analysis that, for independent tasks (Bleuse et al. 2017; Hong and Prasanna 2007) (i.e., there is no data dependence between tasks), if the solution of *task hardening* is given, the solution of *task mapping* can be optimally obtained by employing min–max-weight perfect matching (MMW-PM). Specifically, both DMR- and TMR-based hardening techniques are considered, and we show how to link the problem of *task mapping* with the goal of minimizing the worst-case makespan (i.e., the maximum completion time of all tasks) to the MMW-PM model, and then, based on binary search and Hungarian algorithm (Kuhn 1955), we use an efficient heuristic algorithm [proposed in our previous work (Zhong et al. 2016) to obtain an MMW-PM from a bipartite graph with polynomial time complexity. Besides, as there is a trade-off between cost and time performance (Erbas et al. 2006), we propose a multi-objective memetic algorithm (MOMA) -based task hardening method to obtain a set of solutions with different numbers of cores (i.e., costs), so the designer can choose a solution from the Pareto front according to user preferences, such as cost budget or performance requirement. The key idea of MA (Chen et al. 2011; Wang et al. 2010) is to incorporate problem-specific knowledge into the global search of EAs. Our experimental studies have demonstrated the effectiveness of the proposed method and have shown that by combining the results of MOMA and MOEA we can provide a designer with a highly accurate set of solutions in a reasonable amount of time.

Figure 1 shows an overview of our novel contributions: According to the nested structure of the design of fault-tolerant multi-cores, the proposed MOMA-based *task hardening* can be regarded as the upper-level optimizer for the alternative between DMR and TMR for each task, while the proposed MMW-PM-based *task mapping* can be regarded as the lower-level solver for task-to-core assignments (*for all original tasks and new tasks*) in an optimal way.

The rest of this paper is organized as follows. In Sect. 2, we introduce the adopted problem model. The proposed design method of fault-tolerant multi-cores, including the optimal MMW-PM-based *task mapping* and MOMA-based *task hardening*, is detailed in Sects. 3 and 4. The simulation results and discussions are given in Sect. 5. Finally, Sect. 6 concludes this paper.

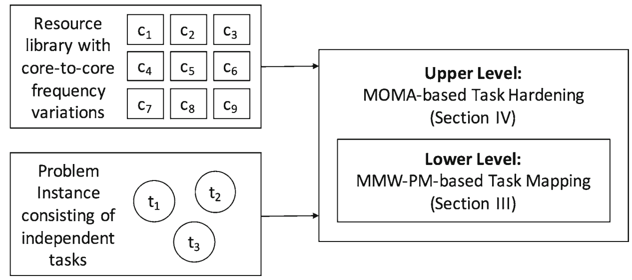


Fig. 1 Overview of the proposed method, illustrating interactions between different contributions

2 Problem definition

2.1 System model

We consider a resource library $C = \{c_1, c_2,\ldots, c_M\}$ consisting of M ISA-compatible RISC cores, which only has single thread per core. Each core c_i has its own instruction and data cache to execute tasks. Due to the performance heterogeneity, e.g., process variations (Herbert et al. 2012; Raghunathan et al. 2013), each core c_i has its own frequency, denoted as f_i, which represents the number of instructions invoked by the core per second. For notational brevity, we index the M cores by a non-decreasing order of the current frequencies, i.e., $f_{\max} = f_1 \geq f_2\ldots \geq f_M = f_{\min}$.

Independent tasks have been used in modeling some practical applications (Bleuse et al. 2017; Hong and Prasanna 2007), e.g., Monte Carlo simulations and computational phylogeny. In these scenarios, each task is to process a fixed amount of source data. The source data of all the tasks initially reside on a single node in the system, which we call the root node.

An instance of the problem is described as a set $T = \{t_1, t_2,\ldots, t_N\}$ of N independent tasks (Bleuse et al. 2017; Hong and Prasanna 2007). Task is the atomic unit executed by the multi-core platform. Due to the functional difference, each task t_i has its own program to be executed by the core, and the number of instructions of the compiled program is denoted as d_i. For notational brevity, we index the N tasks by a non-increasing order of the number of instructions, i.e., $d_{\max} = d_1 \geq d_2\ldots \geq d_N = d_{\min}$. And, the execution time of task t_i on core c_j can be evaluated as $et(t_i, c_j) = \mathrm{CPI}\cdot d_i/f_j$, where CPI represents the clock cycle per instruction.

2.2 Hardening technique

In this paper, both DMR with re-execution (Kang et al. 2015; Pradhan and Vaidya 1994; Ziv and Bruck 1997) and TMR (Chen et al. 2016; Lyons and Vanderkulk 1962) are considered to be against transient faults. For a task hardened by DMR with re-execution, fault detection is achieved by DMR. If a fault is detected, the tasks are executed again by rolling back. For a task hardened by TMR, fault masking is directly achieved by majority voting. Both DMR with re-execution and TMR with major voting can achieve very high reliability even given a high fault rate. For example, if the fault rate of a task on a core is $\lambda = 10^{-6}$ or 10^{-7} (in the unit of #fault/cycles) to realize high fault scenarios as adopted by the related works (Hu et al. 2006; Li et al. 2004), the fault rate of the task hardened by TMR can be evaluated as $\lambda_{\mathrm{TMR}} = \lambda^3 + 3(1-\lambda)\lambda^2 \approx 3 \times 10^{-12}$ or 3×10^{-14}, and the fault rate of the task hardened by DMR with re-execution can be evaluated as $\lambda_{\mathrm{DMR}} = 1 - \{(1-\lambda)^2 + (1-(1-\lambda)^2)((1-\lambda)^2)\} \approx 4 \times 10^{-12}$ or 4×10^{-14}; here, it is assumed that the maximum number of faults during the execution of each task is no more than 1.

Both DMR with re-execution and TMR can achieve very high reliability, and they are characterized by different overheads in cost and time. Obviously, three cores are required in TMR, while only two cores are required in DMR with re-execution. On the other hand, the worst-case execution time (WCET) of a task t_i in TMR mode can be evaluated as $wcet(t_i) = \max\{et(t_{i1}), et(t_{i2}), et(t_{i3})\} + et_v$, where tasks t_{i1}, t_{i2}, and t_{i3} are replicas of task t_i, and e_v is the execution time of major voting, while the WCET of a task t_i in DMR mode can be evaluated as $wcet(t_i) = 2\max\{et(t_{i1}), et(t_{i2})\} + et_r + \text{etc.}$, where tasks t_{i1} and t_{i2} are replicas of task t_i, etc. is the execution time of comparison, and et_r is the execution time of rollback. Since multi-core platform can provide low-overhead communication for comparing and voting, compared with $et(t_i)$, both etc. and et_v are negligible. Therefore, TMR mode is more efficient than DMR mode in terms of WCET.

2.3 Problem statement

Assume we are given a resource library $C = \{c_1, c_2, \ldots, c_M\}$ with different frequencies $f_{\max} = f_1 \geq f_2 \ldots \geq f_M = f_{\min}$, and an instance consisting of independent tasks $T = \{t_1, t_2, \ldots, t_N\}$ with different numbers of instructions $d_{\max} = d_1 \geq d_2 \ldots \geq d_N = d_{\min}$, the goal of this paper is to (1) determine TMR mode or DMR mode for each task, and (2) allocate a core for each task (*including all original tasks and new tasks*), such that (1) the number of cores used, i.e., $K = 2K_{\mathrm{DMR}} + 3K_{\mathrm{TMR}}$, is minimized, where K_{DMR} and K_{TMR} are the number of tasks hardened by DMR and TMR, respectively, and $K_{\mathrm{DMR}} + K_{\mathrm{TMR}} = N$, and (2) the worst-case makespan, i.e., $wcet(T) = \max\{wcet(t_1), wcet(t_2), \ldots, wcet(t_N)\}$, is minimized. It is assumed that we have enough cores in the library so that we do not map two tasks or replicas to the same core in order to fully explore the parallelism of the multi-core platform.

To our knowledge, this is the first work specialized for independent tasks in the design of fault-tolerant multi-cores, as all the previous works (Das et al. 2014; Gan et al. 2012; Khosravi et al. 2014; Pop et al. 2009; Stralen and Pimentel 2012) consider general task sets (e.g., there are data dependences between tasks). As mentioned above, *task mapping* for general task sets is a well-known NP-hard problem. Independent task set, as a special kind of task sets, has been widely applied in modeling many real-world applications (Alazzoni and Down 2008; Bleuse et al. 2017; Cortadella et al. 2006; Hong and Prasanna 2007), but without considering faults. As shown in Sect. 3, under the assumption that the execution modes for all tasks are already known beforehand (Sect. 4), the *task mapping* of independent tasks can be optimized by employing min–max-weight perfect matching (MMW-PM) to minimize the worst-case makespan with polynomial time complexity.

2.4 Optimal task mapping

2.4.1 Greedy mapping algorithm and a motivational example

The *greedy mapping* algorithm (Algorithm 1) is to map tasks to the cores that can be completed as fast as possible. In the following, we provide a motivational example to explain why *greedy mapping* is not good enough for *task mapping*. Suppose that we are given three tasks, i.e., t_1, t_2, and t_3. Task t_1 is executed in TMR mode, while task t_2 and t_3 are executed in DMR mode. Now, we consider the *task mapping* problem to allocate the cores to the tasks for minimizing the worst-case makespan, and the execution times of tasks on the cores are shown in Table 1. For simplicity, et_v, etc., and et_r are supposed to be zeroes here. In this example, we can check all the possible mappings to obtain the optimal result that will be 74.16 ms, where TMR mode task t_1 uses core group $\{c_1, c_2, c_6\}$, while DMR mode tasks t_2 and t_3 use core groups $\{c_3, c_7\}$ and $\{c_4, c_5\}$. By using *greedy mapping* to assign the tasks and cores, the result is 79.08 ms, where TMR mode task t_1 uses core group $\{c_3, c_4, c_7\}$, while DMR mode tasks t_2 and t_3 use core groups $\{c_2, c_5\}$ and $\{c_1, c_6\}$. In the above example, we can observe that the *greedy mapping* strategy is not good enough. As a consequence, it is clear that such a *task mapping* problem requires a better strategy, whereas the straightforward exhaustive search is obviously not feasible in practice with the expected high time complexity.

Table 1 Execution time of tasks on cores (ms)

Execution time	c_1	c_2	c_3	c_4	c_5	c_6	c_7
t_1	44.19	39.14	38.08	38.10	38.80	41.54	29.24
t_2	43.03	38.11	37.08	37.10	37.78	40.45	28.47
t_3	39.54	35.02	34.07	34.09	34.72	37.17	26.16

Algorithm 1: Greedy Mapping for *Task Mapping*

```
Input: the given task hardening solution th
Output: the task mapping solution tm
while there are unmapped tasks do
   select an unhardened task t_i with maximum d_i
   if th_i = 0 %DMR
     select two unallocated cores c_j, c_l with maximum f_j, f_l
     tm_i ← {c_j, c_l}
   else if th_i = 1 %TMR
     select three unallocated cores c_j, c_l, c_k with maximum f_j, f_l, f_k
     tm_i ← {c_j, c_l, c_k}
   end if
   mark task t_i mapped and cores {c_j, c_l} or {c_j, c_l, c_k} allocated
end while
return tm
```

Algorithm 2: Heuristic Algorithm for MMW-PM [60]

```
Input: Weighted bipartite graph: G = (U, V, E, W)
Output: MMW-PM: M
M ← Hungarian algorithm for perfect matching (G)
E' ← Sort E in ascending order of their weights W
low ← 1, high ← |E'|
while low < high
   mid ← ⌊low/2+high/2⌋
   e ← E'(mid)
   G' ← Remove all the edges whose weights are no less than that of e
   M' ← Hungarian algorithm for perfect matching (G')
   if |M'| = |M| then
     M ← M'
     mid ← high
   else
     low ← mid+1
   end if
end
return M
```

2.5 Min–max-weight perfect matching

Given a bipartite graph $G = (U, V, E)$, where U and V are disjoint and all edges in E go between U and V. A matching is a subset of edges $M \in E$, such that for all vertices $v \in U \cup V$, at most one edge of M is incident on v, that is, no two edges share a common vertex. We say that a vertex $v \in U \cup V$ is matched by matching M if some edge in M is incident on v; otherwise, v is unmatched. A perfect matching is a matching which matches all vertices of the graph, that is, every vertex of the graph is incident to exactly one edge of the matching. The problem of finding a perfect matching can be solved by Hungarian algorithm (Kuhn 1955).

Min–Max-weight perfect matching (MMW-PM) is defined on a weighted bipartite graph as a perfect matching, where the maximal weight of the edges in the matching has a minimal value among all perfect matchings of the given graph. One should note that the MMW-PM problem is different from the maximum (or minimum)-weight perfect matching, where the sum of the weights of the edges in the matching has a maximal (or minimal) value among all perfect matchings of the given graph.

It is easy to link the *task mapping* problem to the proposed MMW-PM model. Given a *task hardening* solution, we can construct a new task set as $T' = \{t'_1, t'_2, \ldots, t'_K\}$ by replicating each task in $T = \{t_1, t_2, \ldots, t_N\}$ by 2 (DMR mode) or 3 (TMR mode) one by one, where $K = 2K_{\text{DMR}} + 3K_{\text{TMR}}$, and K_{DMR} and K_{TMR} are the number of tasks hardened by DMR and TMR, respectively. On the other hand, we can obtain the resource allocation, $C' = \{c'_1, c'_2, \ldots, c'_K\}$, by selecting the first K cores with the highest frequencies from $C = \{c_1, c_2, \ldots, c_M\}$. Then, a bipartite graph can be built as $G' = (T', C', E')$, and E' go between each pair of t'_i and c'_j. The associated weight of each edge $\left(t'_i, c'_j\right)$ can be calculated according to: 1) $w\left(t'_i, c'_j\right) = et\left(t'_i, c'_j\right) + et_v$, if task t'_i is executed in TMR mode, or 2) $w\left(t'_i, c'_j\right) = 2et\left(t'_i, c'_j\right) + et_r + et_c$, if task t'_i is executed in DMR mode. Then, for a MMW-PM M' from G', the matchings in M' represent the mappings from task replicas to cores, and the associated maximum weight of the edges is the worst-case makespan.

It is notable that the associated weight $w\left(t'_i, c'_j\right)$ of each edge is not the WCET of t'_i on c'_j, because comparing and voting are invoked until all replicas of the same task are completed. Actually, there is no way to know the execution time of other replicas of t'_i beforehand, because we do not know the mappings at all. But, the worst-case makespan can be evaluated as:

$$
\begin{aligned}
& wcet(T) = \max\{wcet(t_1), wcet(t_2), \ldots\} \\
& = \max\{\max\{et(t_{11}), et(t_{12}), et(t_{13})\} \\
& \quad + et_v \text{ or } 2\max\{et(t_{11}), et(t_{12})\} + et_r \\
& \quad + et_c, \max\{et(t_{21}), et(t_{22}), et(t_{23}) \\
& \quad + et_v \text{ or } 2\max\{et(t_{21}), et(t_{22})\} + et_r + et_c, \ldots\} \\
& = \max\{\{et(t_{11}) + et_v, et(t_{12}) + et_v, et(t_{13}) + et_v\} \text{ or } \{2et(t_{11}) \\
& \quad + et_r + et_c, 2et(t_{12}) + et_r \\
& = + et_c\}, \{et(t_{21}) + et_v, et(t_{22}) + et_v, et(t_{23}) + et_v\} \text{ or } \{2et(t_{21}) \\
& \quad + et_r + et_c, 2et(t_{22}) + et_r \\
& \quad + et_c, \ldots\}\max\left\{w\left(t'_1\right), w\left(t'_2\right), \ldots\right\}.
\end{aligned}
$$

Therefore, we can obtain the real worst-case makespan of T by employing the above weighting strategy.

2.6 MMW-PM heuristic

The above section shows how to link the *task mapping* problem to the MMW-PM model. A naive way to find an MMW-PM (i.e., the optimal mapping) is to find all perfect matchings in the given bipartite graph first and then select the one whose maximal edge weight is minimal. Instead of using such an enumeration method, we use an efferent heuristic algorithm [proposed in our previous work (Zhong et al. 2016) for MMW-PM problem with low time complexity.

Given an undirected weighted bipartite graph $G = (U, V, E, W)$, where U and V are disjoint and all edges in E go between U and V. At first, we sort the edges in G in ascending order of their weights. Our heuristic is to select an edge e in G and remove all the edges whose weights are larger than that of e in G, while a perfect matching method (i.e., Hungarian algorithm (Kuhn 1955)) is used to check whether a perfect matching exists. The framework of the heuristic method is iterative based on binary search, as shown in Algorithm 2 (Zhong et al. 2016). The algorithm starts with an initial perfect matching M obtained by the Hungarian algorithm (line 1), and then we have E' by sorting E in ascending order according to their weights (line 2). At each iteration, an edge e in E' is selected by binary search (lines 5 and 6) and a new graph G' is obtained through removing all the edges with larger weights in G (line 7), and then we can obtain a new matching M' by running the Hungarian algorithm on G' (line 8). If the cardinality of M' equates to that of M, i.e., M' is a perfect matching as well, we update M to M' (line 10) and set *mid* as *high* (line 11), otherwise, we increase *low* by 1 (line 13). The final solution M is returned until the loop terminates (line 16).

Obviously, Algorithm 2 (Zhong et al. 2016) would return a perfect matching of the input graph, as the resulting matching M has the same cardinality as the initialized M obtained from the input graph (line 1). Besides, the edges in G are removed according to the ascending order of their weights, so the resulting matching M satisfies that the maximal weight of the edges in M has a minimal value among all perfect matchings of the input graph. Therefore, Algorithm 2 can indeed find an MMW-PM from the given graph with the advantage of a high efficiency over the enumeration method. Given an undirected weighted bipartite graph $G = (U, V, E, W)$, with $|U| = |V| = n$ ($n = K$ in our case), the time complexities of Hungarian algorithm and binary search are $O(n^3)$ and $O(\log n)$, respectively. Therefore, the time complexity of Algorithm 2 is $O(n^3 \log n)$.

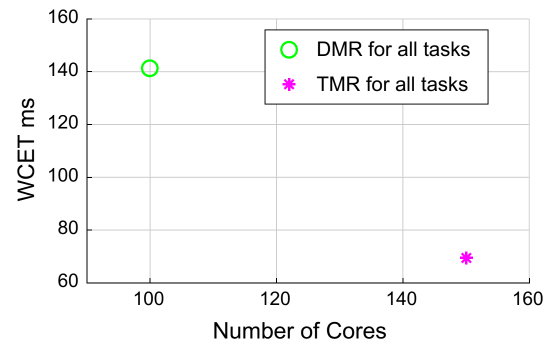


Fig. 2 An example to show the trade-offs in task hardening

3 Multi-objective redundancy hardening

3.1 Trade-offs in task hardening

As analyzed in Sect. 2, DMR mode and TMR mode are characterized by different overheads in cost and time, and there is a trade-off in determining DMR or TMR for each task. Figure 2 shows a simple example of the trade-offs in *task hardening* for a problem instance, where the green circle represents the solution that all tasks are hardened by DMR, and the magenta star represents the solution that all tasks are hardened by TMR. “DMR for all tasks” uses the least number of cores, while “TMR for all tasks” achieves the best performance in terms of worst-case makespan. In order to provide the designer with a set of solutions with different trade-offs in cost and performance, we model the problem as a multi-objective optimization problem. Specifically, in terms of cost, the objective is to minimize the number of cores used, while in terms of performance, the objective is to minimize the worst-case makespan, so we have:

Algorithm 3: MOEA-based *Task Hardening*
N: the number of tasks
M: the number of cores
PS: the population size of task hardening solutions
TH: the parent population of task hardening solutions
NTH: the offspring population of task hardening solutions
P_c: the probability of crossover
P_m: the probability of mutation

Input: resource library $C = \{c_1, c_2, \ldots, c_M\}$ and problem instance $T = \{t_1, t_2, \ldots, t_N\}$
Output: a set of nondominated *task hardening* solutions *th*s
$TH = \{th^i\}$, $th^i = \{th_1, th_2, \ldots th_N\} \in \{0, 1\}^N$, $i = \{1, 2, \ldots PS\}$
evaluate $Fitness_{TH} = \{Object_1{}^i, Object_2{}^i\}$ based on Eq. (1) and Eq. (2), $i = \{1, 2, \ldots PS\}$
repeat
  for $i = 1$ **to** PS **do** // evolve the *task hardening* population
    $(th^j, th^k) \leftarrow$ **Selection for Reproduction** (TH)
    $nth^i \leftarrow$ **Crossover** (th^j, th^k, P_c)
  end for
  for $i = 1$ **to** PS **do**
    $nth^i \leftarrow$ **Mutation** (nth^i, P_m)
  end for
  evaluate $Fitness_{NTH} = \{Object_1{}^i, Object_2{}^i\}$ based on Eq. (1) and Eq. (2), $i = \{1, 2, \ldots PS\}$
  $TH \leftarrow$ **Nondominated Sorting** $(TH, NTH, Fitness_{TH}, Fitness_{NTH})$
until maximum number of fitness evaluations is reached

$$Object_1 : K = 2K_{DMR} + 3K_{TMR} \tag{1}$$

$$Object_2 : wcet(T) = \max\{wcet(t_1), wcet(t_2), \ldots, wcet(t_N)\}. \tag{2}$$

3.2 MOEA-based task hardening

In recent years, MOEAs have been successfully applied to many kinds of multi-objective optimization problems (Shen and Yao 2015; Wang et al. 2015). While there is a wide variety of MOEAs, in this paper, we just consider the most popular one, i.e., nondominated sorting genetic algorithm II (NSGA-II) (Deb et al. 2002). The key ideas of NSGA-II include a fast nondominated sorting approach with low computational complexity and a novel selection process that creates a mating pool by combining the parent and offspring populations and selecting the best (with respect to fitness and spread) solutions. Genetic algorithm (GA) is adopted in the NSGA-II framework to work as the evolutionary engine for the population. GA generates solutions to the optimization problem using operations inspired by Darwinian principles of natural evolution, such as recombination (crossover), mutation, and selection. The detailed design of the elementary steps for evolving the population of solutions of *task hardening* is given as follows.

(1) Encoding

In GA, the solution to the optimization problem is encoded in a chromosome as a set of parameters. In our implementation, each chromosome in the *task hardening* population is represented by a N-dimensional 0–1 vector $th = \{th_1, th_2, \ldots th_N\} \in \{0, 1\}^N$, where N is the number of tasks. Each position of the vector describes the fault-tolerant technique that assigned to the task, i.e., $th_i = 0$ indicates that task t_i is hardened by DMR with re-execution and $th_i = 1$ indicates that task t_i is hardened by TMR.

(2) Crossover

In GA, crossover is a genetic operator that recombines more than one (generally two) parent chromosomes to produce one or two child chromosomes from them. In our implementation, we use one-point crossover, the simplest one, i.e., a single crossover point on both parent vectors is selected first, and then all data beyond that point in either vector are swapped between the two parent vectors, the resulting vectors are the child chromosomes. In this way, the applied fault-tolerant techniques are mixed for the *task hardening* solutions.

(3) Mutation

In GA, mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state to maintain the genetic diversity of the population. In general, mutation operators involve a probability that an arbitrary gene in a chromosome would be changed from its original state, and this probability should be set low. In our implementation, we use flip mutation for the 0–1 vector, i.e., the value of the chosen gene is inverted. In this way, the applied fault-tolerant technique is changed for the task in the *task hardening* solution.

(4) Selection

Selection occurs two times during each generation in GA. In our implementation, selection for reproduction is performed before the crossover operator is applied, which is based on a purely random basis without bias to filter any individual, and selection for survival is performed according to NSGA-II, i.e., a fast nondominated sorting approach and a novel selection process with the consideration of both fitness and spread.

The outline of the *MOEA-based hardening* is given in Algorithm 3, where PS, P_c, and P_m indicate the population size, probability of crossover, and probability of mutation, respectively. The algorithm starts with a population TH consisting of PS random individuals for *task hardening* (line 1). The fitness in terms of cost, i.e., $Object_1$, is evaluated according to Eq. (1), and the fitness in terms of performance, i.e., $Object_2$, is evaluated based on the proposed *MMW-PM-based mapping* (Algorithm 2) and Eq. (2) (line 2). During each generation, TH is evolved and the population of PS individuals generates PS children through the crossover operation (lines 4–7) and the mutation operation (lines 8–10). Then, the offspring NTH are evaluated (line 11) and used to update the current population (line 12) based on the fast nondominated sort. When the given maximum number of fitness evaluations is reached, the algorithm stops (line 13).

3.3 MOMA-based task hardening

As shown in the following experiments (Sect. 5), it is very difficult to obtain the whole Pareto front using the proposed *MOEA-based hardening* method so that we cannot provide well-distributed solutions for the designer to choose. In particular, it is hard to minimize $Object_2$ by NSGA-II, so the solutions are all partiality located at the range where $Object_1$ is minimized. This is because $Object_2$ involves a complex nonlinear mapping from decision variables to fitness value (i.e., using *MMW-PM-based mapping*), while $Object_1$ is just a linear combination of the variables (i.e., Eq. 1), and the select pressure on $Object_2$ is too weak in the adopted NSGA-II framework.

In order to incorporate bias in the search for $Object_2$, we propose a problem-specific local search operator, and it can be regarded as a kind of memes in the case of MAs (Rubio-Largo et al. 2016; Tersi et al. 2015; Yuan and Xu 2015). In combinatorial optimization, local search operators generally work in the form of heuristics that are customized to a specific problem. In our implementation, the key idea of the heuristic is inherited from the previous greedy reassignment local search (GRLS) operators for *logic mapping* (Yuan et al. 2016; Yuan et al. 2014), which is to reassign the gene values of part of the parent chromosome by taking advantage of the greedy information extracted from the problem instance. In this paper, this idea is extended for *task hardening*.

As analyzed in Sect. 2, TMR mode is more efficient than DMR mode; it is expected that a task can be completed earlier if a task is executed in TMR mode. The GRLS operator designed for *task hardening* tries to flip the hardening technique from DMR to TMR for the given task t_i with larger d_i. It is expected to reduce the worst-case makespan, i.e., $Object_2$. In order to release the time overhead added to the iterative process of GA, the time complexity of the operator should be as low as possible. In fact, for each task, the priority list of tasks in terms of d_i can be sorted in advance; thus, the greedy information of the problem instance only needs to be evaluated once before the optimization process.

Since the operator is designed to be complementary to the stochastic search of GA, the incorporation of the operator should maintain the randomness as well. Besides, an operator with strong greediness will weaken the stochastic nature of GA and improve the risk of convergence to the local optima. Therefore, care should be taken when setting the strength of the GRLS operator in achieving the best search performance. In our implementation, for a given solution, a control parameter is introduced to limit the number of tasks that will be applied to by the GRLS operator. For example, $N \cdot \mu$ tasks are randomly selected, where $0 \leq \mu \leq 1$ is defined as the greedy strength factor that provides a flexible control on randomness or greediness of the GRLS operator.

The outline of the proposed GRLS operator for *task hardening* is given in Algorithm 4. $N \cdot \mu$ tasks are randomly selected and marked as unhardened (line 1). In the loop (lines 2–6), an unhardened task t_i with maximum d_i is selected (line 3); if t_i is hardened by DMR, then change DMR to TMR, and the loop terminates (line 4). By applying Algorithm 4 to each generated solution during the *MOEA-based hardening* algorithm, we obtain the *MOMA-based hardening* algorithm.

Algorithm 4: GRLS for *Task Hardening*

```
Input: the given task hardening solution th
Output: new task hardening solution th
01: randomly select N·μ tasks, and mark them unhardened
02: while there are unhardened tasks do
03:   select an unhardened task t_i with maximum d_i
04:   if th_i = 0 then th_i = 1; break; end if
05:   mark task t_i hardened
06: end while
07: return th
```

4 Simulations and discussion

In this section, the performance of the proposed method is experimentally investigated. First, we show that the proposed *MMW-PM-based mapping* consistently outperforms *greedy mapping*, and we test the impact of control parameter μ on the performance of the proposed *MOMA-based hardening*. Then, compared with *MOEA-based hardening*, the effectiveness of *MOMA-based hardening* is verified by extensive experiments, especially on large-scale benchmarks. At the final, we show that by combining the results of *MOMA-based hardening* and *MOEA-based hardening* we can provide the designer with a highly accurate set of solutions in a reasonable amount of time.

In the experiments, a large set of benchmarks with different scales are synthesized, e.g., $N = 10$, 30, 50, and 100 that indicates the number of tasks in the problem instances, and $M > 3\,N$ accordingly that indicates the number of cores in the resource library. We generate four test instances for each case. The number of instructions (d) of the compiled tasks ranges from 10^7 to 5×10^7 and randomly determined, and the value of CPI is assumed to be 1.5. The frequencies (f) of cores are normally distributed with mean 1 GHz and standard deviation 0.1 GHz. The voting time (et_v) involved in TMR, the comprising time (etc.) involved in DMR, and the recovering time (et_r) involved in re-execution are assumed to be 2 ms, 2 ms, and 6 ms, respectively.

GA is used as the evolutionary search engine, and the parameters of population size (PS), probability of crossover (P_c), and probability of mutation (P_m) are set as $PS = 2\,N$, $P_c = 0.8$, and $P_m = 0.8/N$ experimentally. The maximum number of generations is set to be 50 to limit the runtime (thus the number of function evaluations is 100 N), and in most cases, both MOMA and MOEA have converged to good solutions. Both generational distance (GD) and inverted generational distance (IGD) are used as performance metrics for the proposed *MOMA-based hardening* and *MOEA-based hardening*. GD is defined as the mean distance in the objective space from each obtained solution to its nearest Pareto optimal solution, while IGD is defined as the mean distance in the objective space from each Pareto optimal solution to its nearest obtained solution. GD provides a measure of proximity of the nondominated solutions in the objective space with respect to the Pareto optimal front, while IGD provides a measure of both proximity and diversity of the nondominated solutions in the objective space with respect to the Pareto optimal front; the smaller the value is, the better the Pareto optimal front is approximated. As customary, the nondominated solutions obtained by multiple runs of both MOMA and MOEA with larger populations and more generations are assumed as the Pareto optimal set, which is the reference set of GD and IGD. We implement the algorithms in MATLAB. All the experiments are performed on a 3.2 GHz Intel Core i5-6500 quad-core platform with 8 GB memory. However, all the tested algorithms are implemented as monolithic processes and no CPU core parallelism is exploited.

4.1 MMW-PM-based mapping versus greedy mapping

As analyzed in Sect. 3, if the solution of *task hardening* is given, the proposed *MMW-PM-based mapping* can obtain the optimal *task mapping* solution. Figure 3 shows the average worst-case makespan (WCET) of *greedy mapping* and *MMW-PM-based mapping* on test instances of different scales. In these simulations, since it is impossible to evaluate the algorithms with all possible *task hardening* solutions, each bar in the presented figures is obtained by averaging the results through multiple runs (e.g., 2^9, 2^{10}, 2^{11}, and 2^{12} for $N = 10$, 30, 50, and 100, respectively). As shown in Fig. 3, we can observe that the proposed *MMW-PM-based mapping* consistently outperforms *greedy mapping* on all cases, and the average improvements are 3.40%, 3.74%, 2.74%, and 2.54% for different problem scales ($N = 10$, 30, 50, and 100, respectively).

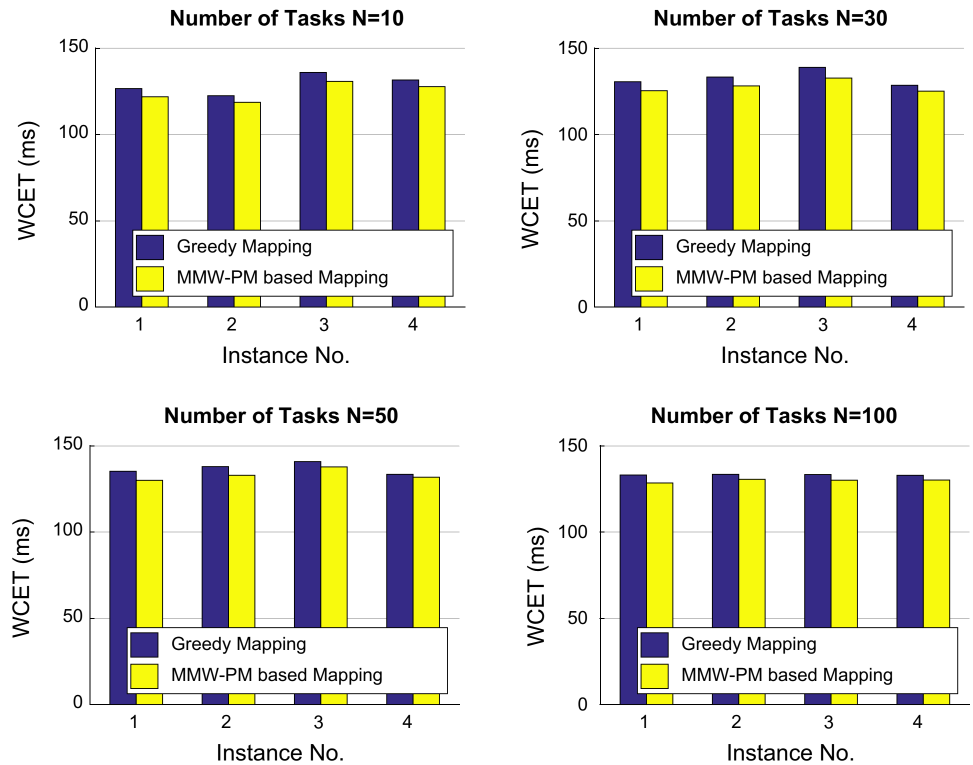
4.2 Sensitivity of parameter μ

Parameter μ controls the randomness or greediness of the GRLS operator; therefore, it provides a balance between the two conflicting objectives. Figure 4 shows the evolutionary curves in terms of IGD for MOMAs with different values of μ on middle-size test instances of $N = 50$. To make the simulations more convincing, each group result is averaged over 30 runs. As shown in the figure, the larger the values of μ, the faster the MOMA approximates the Pareto optimal front in terms of IGD. But, a very large μ, i.e., $\mu = 0.9$, results in bad performances at the final stage; this is because such strong greediness disturbs the stochastic search of GA too much. On the contrary, a very small μ, i.e., $\mu = 0.1$, cannot accelerate the approximation to the Pareto front significantly, but it indeed leads to good results if granted a long runtime. Therefore, a moderate value of μ is preferred, e.g., $\mu = 0.3$ or 0.5 or 0.7, and the performance of *MOMA-based hardening* is not very sensitive to the values of μ in this range, i.e., from 0.3 to 0.7. We use $\mu = 0.5$ in the following simulations.

4.3 MOMA-based hardening versus MOEA-based hardening

Figures 5, 6, 7, and 8 show the evolutionary curves of GD and IGD for *MOMA-based hardening* and *MOEA-based*

Fig. 3 Average worst-case makespan of *greedy mapping* and *MMW-PM-based mapping*



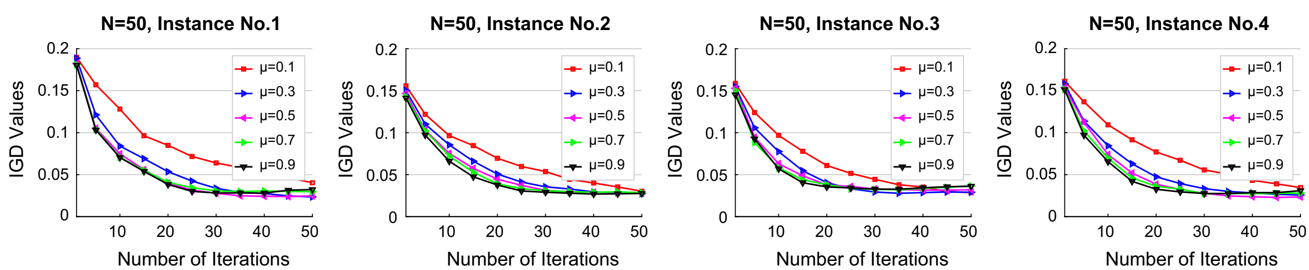


Fig. 4 Evolutionary curves of IGD for MOMAs with different values of μ on test instances of $N = 50$ scale

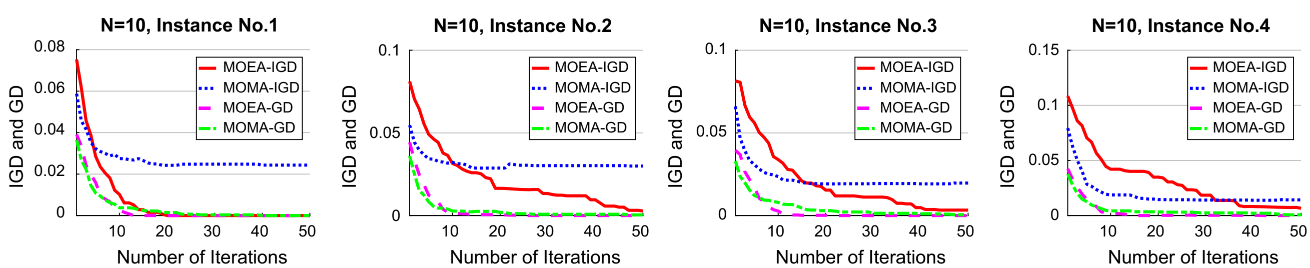


Fig. 5 Evolutionary curves of GD and IGD for MOEA and MOMA on test instances of $N = 10$ scale

hardening on different test instances of different scales ($N = 10$, 30, 50, and 100, respectively). To make the simulations more convincing, each group result is averaged over 30 runs. It can be seen that: (1) for small-scale problem (e.g., $N = 10$), *MOEA-based hardening* performs better than *MOMA-based hardening*, and (2) for large-scale problems (e.g., $N = 30$, 50 and 100), the mean IGD values obtained by *MOMA-based hardening* are much lower than those obtained by *MOEA-based hardening* on most test instances, which is an indication of better approximation to

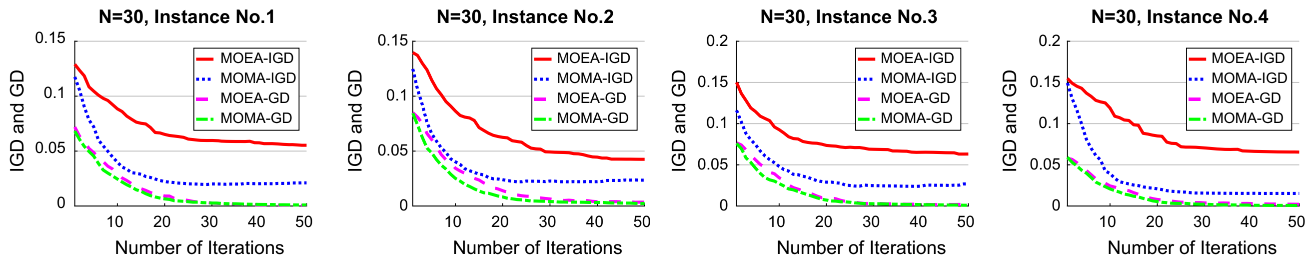


Fig. 6 Evolutionary curves of GD and IGD for MOEA and MOMA on test instances of $N = 30$ scale

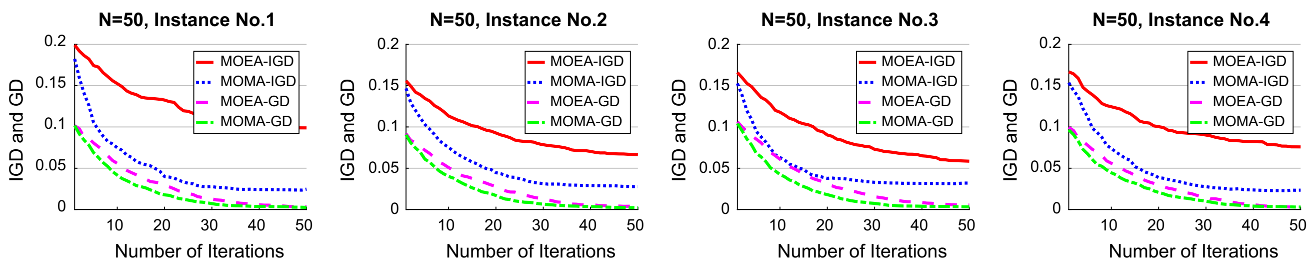


Fig. 7 Evolutionary curves of GD and IGD for MOEA and MOMA on test instances of $N = 50$ scale

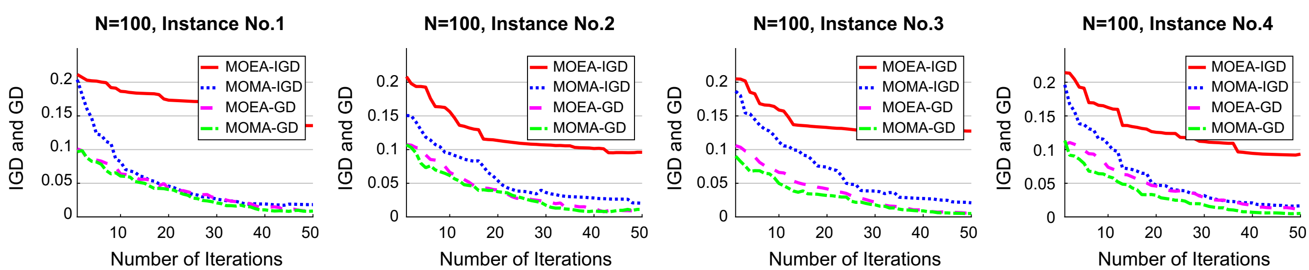


Fig. 8 Evolutionary curves of GD and IGD for MOEA and MOMA on test instances of $N = 100$ scale

the Pareto optimal front. The maximum reductions on IGD values are about 82%, 75%, and 86%, and the average reductions are about 64%, 70%, and 80% for different problem scales ($N = 30$, 50, and 100, respectively). The better approximation can be attributed to the introduction of the problem-specific knowledge, so the effectiveness of the proposed GRLS for *task hardening* is demonstrated.

We have performed statistical tests for the GD and IGD values of MOEA and MOMA on each benchmark instance. A two-tailed t test is conducted with a null hypothesis, stating that there is no difference between two algorithms in comparison. The null hypothesis is rejected if the p value is smaller than the significance level $\alpha = 0.05$. We find that (1) there is no significant difference between the GD values of MOEA and MOMA on all test instances, (2) the IGD values of MOEA are statistically better than those of MOMA on small-scale test instances ($N = 10$), and (3) the IGD values of MOMA are statistically better than those of MOEA on large-scale test instances ($N = 30$, 50, and 100).

4.4 A comprehensive comparison

Figures 9, 10, 11, and 12 show the Pareto fronts obtained by multiple runs (e.g., 5 runs each) of both *MOMA-based hardening* and *MOEA-based hardening* on test instances of different scales ($N = 10$, 30, 50, and 100, respectively). In each figure, the green circle represents the all "0" solution, i.e., $th = \{0\}^N$, that means all the tasks are hardened by DMR, while the magenta star represents the all "1" solutions, i.e., $th = \{1\}^N$, that means all the tasks are hardened by TMR. Obviously, all "0" solutions minimize the number of cores according to Eq. (1), while all "1" solutions can achieve the best performance in terms of worst-case makespan. As shown in the figures, the near "0" solutions can be obtained by using *MOEA-based hardening* method, and we can approach the best performance with fewer cores by using *MOMA-based hardening* method. Another observation is that the results obtained by *MOEA-based hardening* are all partiality located at the area where $Object_1$ (number of cores) is minimized, especially for large-scale problems (e.g., $N = 50$ and 100). As analyzed

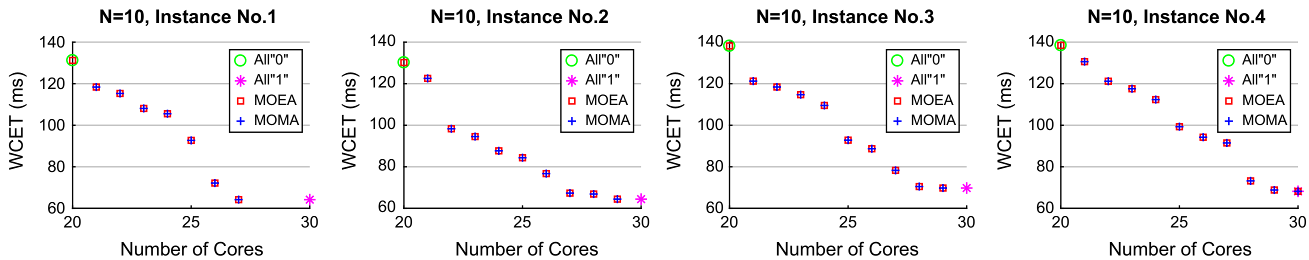


Fig. 9 Pareto front obtained by multiple runs of MOEA and MOMA on test instances of $N = 10$ scale

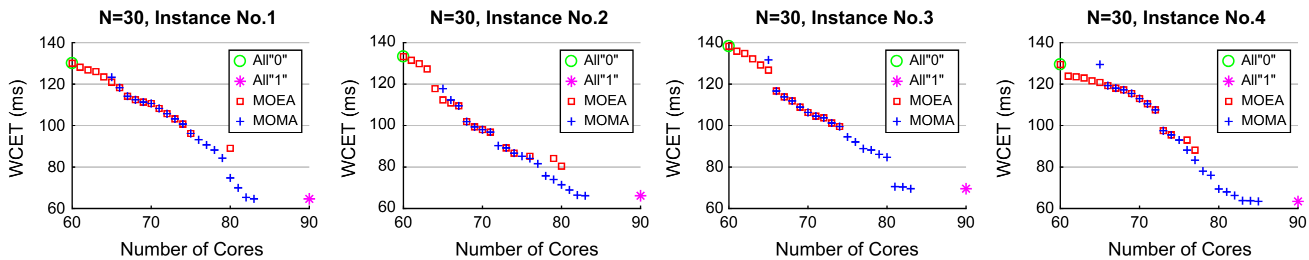


Fig. 10 Pareto front obtained by multiple runs of MOEA and MOMA on test instances of $N = 30$ scale

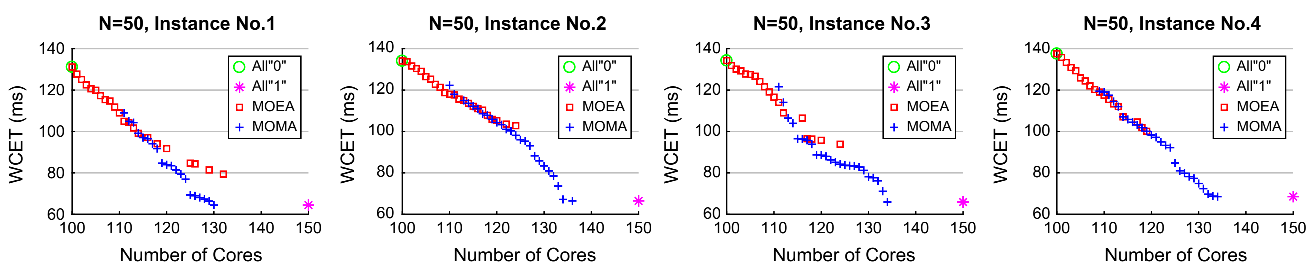


Fig. 11 Pareto front obtained by multiple runs of MOEA and MOMA on test instances of $N = 50$ scale

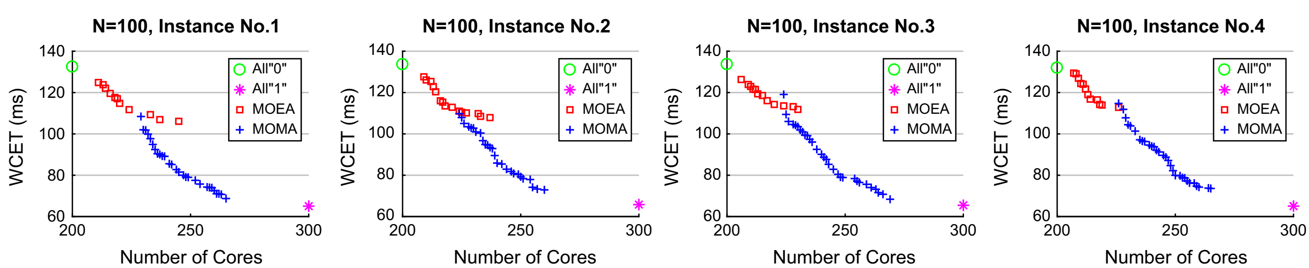


Fig. 12 Pareto front obtained by multiple runs of MOEA and MOMA on test instances of $N = 100$ scale

above, this is because $Object_2$ (WCET) involves a complex nonlinear mapping from decision variables to fitness value, while $Object_1$ is just a linear combination of the values; the select pressure on $Object_2$ is too weak in the adopted NSGA-II framework. By incorporating the proposed GRLS operator, we can see *MOMA-based hardening* can cover this absent area; this is because the proposed operator has bias for minimizing WECT by using more space redundancy (from DMR to TMR). We can obtain a well-distributed Pareto front by combining the solutions of both methods; thus, we can provide the designer with a highly accurate set of solutions.

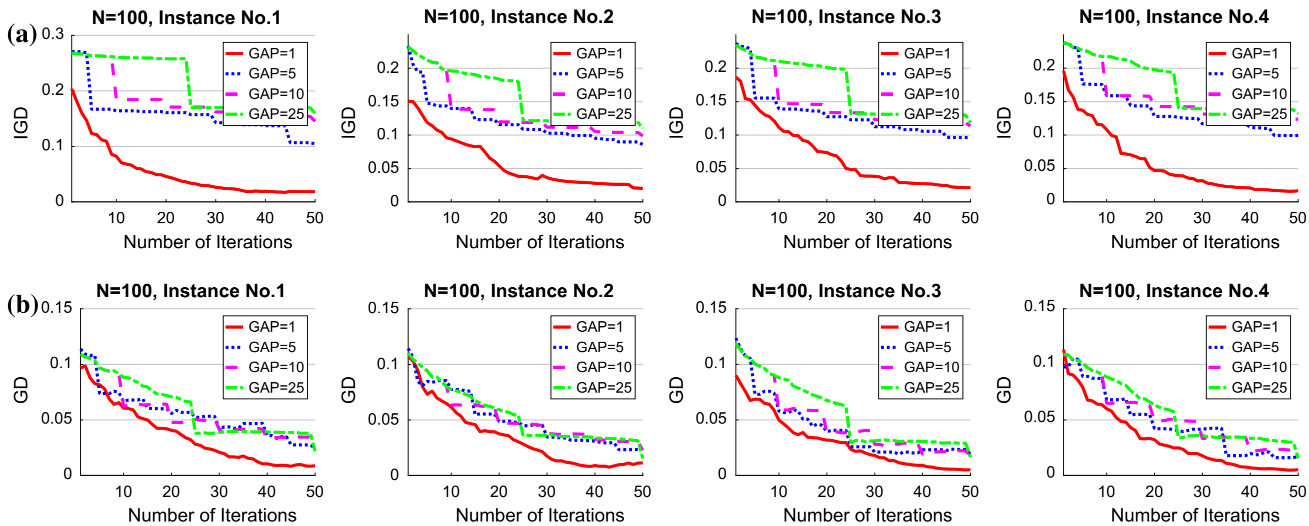


Fig. 13 **a** Evolutionary curves of GD values for MOMA with hybrid fitness evaluation on test instances of N = 100 scale. **b** The evolutionary curves of IGD values for MOMA with hybrid fitness evaluation on test instances of N = 100 scale

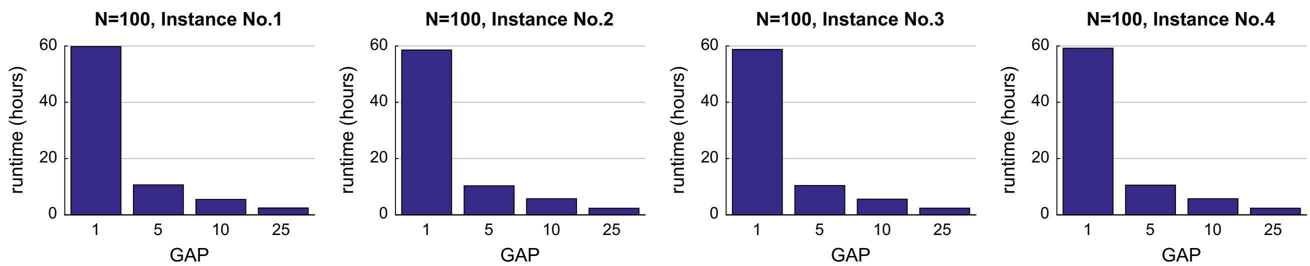


Fig. 14 Runtime of MOMA with hybrid fitness evaluation on test instances of N = 100 scale

4.5 Hybrid fitness evaluation

As the time complexity ($O(n^3 log^n)$) of the Hungarian-based optimal MMW-PM method (Algorithm 2) is high, the whole EA process is very time-consuming for large-scale problems, e.g., more than 2 days for n = 100 test instances. As a remedy, we propose to use hybrid fitness evaluation method to balance the accuracy and time efficiency of fitness evaluations. The key idea is to combine the optimal and greedy MMW-PM methods in the fitness evolution process (Jin et al. 2002; Jin 2005). Specifically, we use single optimal MMW-PM evaluation for the whole population in every Δ iterations, where Δ is called exact evaluation gap here. A few different values of exact evaluation gap Δ = 1, 5, 10, and 25, are tested as shown in Figs. 13 and 14, where Δ is marked as "GAP." The results show that the runtime can be reduced significantly if the preference degradation is acceptable by the designers, e.g., Δ = 5.

After obtaining the Pareto front, it is an open problem of how to make the final decision. From the point of view of multi-core system design, the designer can choose a solution from the Pareto front based on the available hardware resources. For example, if the required resources can be satisfied, the designer can implement a high-performance multi-core system with a low hardware cost. Alternatively, the designer could select the knee point on the Pareto front. It is important to note that the approximate Pareto front found by our algorithm can inform the designer about various trade-offs among conflicting objectives, which is essential in making an informed final decision.

5 Conclusion

This paper aims at jointly *task hardening* and *task mapping* for independent tasks on heterogeneous multi-core systems in order to achieve low cost and high performance. In order to minimize the worst-case makespan with optimal *task mapping*, we show that the mapping problem can be modeled as a min–max-weight perfect matching problem. Then, we propose a polynomial time complexity heuristic algorithm that works in a binary search framework and employs Hungarian algorithm as a subroutine. As there is a trade-off between the cost and performance in *task hardening*, we model the *task hardening* problem as a multi-objective optimization problem. Since a MOEA can find a partial Pareto front only, we propose a problem-specific local search operator and incorporate it in the MOEA framework. As shown by our simulation results, the proposed MOMA method can cover the missing part of the Pareto front, so we can obtain the entire Pareto front by combining the solutions of MOEA and MOMA.

This work considers independent tasks (or parallel tasks); we will extend this work to the tasks with data dependents in our future work, e.g., sequential tasks. Besides, we will consider dynamic task mapping, where the frequencies of cores change with heat, etc.

Acknowledgements This work was supported by the National Key R&D Program of China (Grant No. 2017YFC0804002), the Royal Society (NA160545), the National Natural Science Foundation of China (Grant Nos. 61503357 and 617611360), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X386), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant Nos. ZDSYS201703031748284, JCYJ20170307105521943, JCYJ20170817112421757 and JCYJ20180504165652917), and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest

Human and animal rights This article does not contain any studies with human participants or animals.

References

Alazzoni I, Down DG (2008) Linear programming-based affinity scheduling of independent tasks on heterogeneous computing Systems. IEEE Trans Parallel Distrib Syst 19(12):1671–1682

Baumann RC (2005) Radiation-induced soft errors in advanced semiconductor technologies. IEEE Trans Device Mater Reliab 5(3):305–316

Bleuse R, Hunold S, Kedad-Sidhoum S, Monna F, Mounie G, Trystram D (2017) Scheduling independent moldable tasks on multi-cores with GPUs. IEEE Trans Parallel Distrib Syst 28:9. https://doi.org/10.1109/tpds.2017.2675891

Bolchini C, Miele A (2013) Reliability-driven system-level synthesis for mixed-critical embedded systems. IEEE Trans Comput 62(12):2489–2502

Bolchini C, Miele A, Pilato C (2011) Combined architecture and hardening techniques exploration for reliable embedded system design. In: Proceedings of the Great Lakes symposium on VLSI, 2011, pp 301–306

Bowman K, Duvall S, Meindl J (2002) Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. IEEE J Solid State Circuits 37(2):183–190

Chen X, Ong Y-S, Lim M-H, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607

Chen K-H, Chen J-J, Kriebel F, Rehman S, Shafique M, Henkel J (2016) Task mapping for redundant multithreading in multi-cores with reliability and performance heterogeneity. IEEE Trans Comput 65(11):3441–3455

Constantinescu C (2003) Trends and challenges in VLSI circuit reliability. IEEE Micro 23(4):14–19

Cortadella J, Kondratyev A, Lavagno L, Passerone C, Watanabe Y (2006) Quasi-static scheduling of independent tasks for reactive systems. IEEE Trans Comput Aided Des Integr Circuits Syst 24(10):1492–1514

Das A, Kumar A, Veeravalli B, Bolchini C, Miele A (2014) Combined DVFS and mapping exploration for lifetime and soft-error susceptibility improvement in MPSoCs. In: Proceedings of the design automation and test in Europe conference and exhibition 2014, pp 1–6

Dave B, Jha N (1999) COFTA: hardware-software co-synthesis of heterogeneous distributed embedded systems for low overhead fault tolerance. IEEE Trans Comput 48(4):417–441

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

Dighe S, Vangal SR, Aseron P, Kumar S, Jacob T, Bowman KA, Howard J, Tschanz J, Erraguntla V, Borkar N, De V, Borkar S (2011) Within-die variation-aware dynamic voltage- frequency-scaling with optimal core allocation and thread hopping for the 80-core teraflops processor. IEEE J Solid-State Circuits 46(1):184–193

Ebi T, Faruque M, Henkel J (2009) Tape: thermal-aware agent based power economy multi/many-core architectures. In: Proceedings of the international conference on computer-aided design, pp 302–309

Erbas C, Ceraverbas S, Pimentel AD (2006) Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. IEEE Trans Evol Comput 10(3):358–374

Gan J, Pop P, Gruian F, Madsen J (2012) Robust and flexible mapping for real-time distributed applications during the early design phases. In: Proceedings of the design, automation test on European conference exhibition, pp 935–940

Glaß M, Lukasiewycz M, Streichert T, Haubelt C, Teich J (2007) Reliability-aware system synthesis. In: Proceedings of the design, automation test on European conference exhibition, pp 409–414

Henkel J, Bauer L, Dutt N, Gupta P, Nassif S, Shafique M, Tahoori M, Wehn N (2013a) Reliable on-chip systems in the nano-era: lessons learnt and future trends. In: Proceedings of the 50th annual design, automation conference, pp 1–10

Henkel J, Narayanan V, Parameswaran S, Teich J (2013b) Run-time adaption for highly-complex multi-core systems. In: Proceedings of the international conference on hardware/software codesign and system synthesis, pp 1–8

Herbert S, Garg S, Marculescu D (2012) Exploiting process variability in voltage/frequency control. IEEE Trans Very Large Scale Integr Syst 20(8):1392–1404

Hong B, Prasanna VK (2007) Adaptive allocation of independent tasks to maximize throughput. IEEE Trans Parallel Distrib Syst 18(10):1420–1435

Hu J, Wang S, Ziavras S (2006) In-register duplication: exploiting narrow-width value for improving register file reliability. In:

Proceedings of the international conference on dependable systems and networks, pp 281–290
Huang J, Blech J, Raabe A, Buckl C, Knoll A (2011) Analysis and optimization of fault-tolerant task scheduling on multiprocessor embedded systems. In: Proceedings of the international conference on hardware/software codesign and system synthesis, pp 247–256
Huang J, Huang K, Raabe A, Buckl C, Knoll A (2012) Towards fault-tolerant embedded systems with imperfect fault detection. In: Proceedings of the 49th annual design, automation conference, pp 188–196
Izosimov V, Polian I, Pop P, Eles P, Peng Z (2009) Analysis and optimization of fault-tolerant embedded systems with hardened processors. In: Proceedings of the design, automation test European conference exhibition, pp 682–687
Jahn J, Faruque M, Henkel J (2011) Carat: context-aware runtime adaptive task migration for multi core architectures. In: Proceedings of the design, automation test European conference exhibition, pp 1–6
Jhumka A, Klaus S, Huss S (2005) A dependability-driven system-level design approach for embedded systems. In: Proceedings of the design, automation test on European conference exhibition, pp 372–377
Jin Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. Soft Comput 9(1):3–12
Jin Y, Olhofer M, Sendhoff B (2002) A framework for evolutionary optimization with approximate fitness functions. IEEE Trans Evol Comput 6(5):481–494
Kandasamy N, Hayes J, Murray B (2003) Transparent recovery from intermittent faults in time-triggered distributed systems. IEEE Trans Comput 52(2):113–125
Kang S-H, Yang H, Kim S, Bacivarov I, Ha S, Thiele L (2014a) Reliability-aware mapping optimization of multi-core systems with mixed-criticality. In: Proceedings of the design, automation test European conference exhibition, pp 1–4
Kang S-H, Yang H, Kim S, Bacivarov I, Ha S, Thiele L (2014b) Static mapping of mixed-critical applications for fault-tolerant MPSOCS. In: Proceedings of the 51st annual design automation conference, pp 1–6
Kang S-H, Park H-W, Kim S, Oh H, Ha S (2015) Optimal checkpoint selection with dual-modular redundancy hardening. IEEE Trans Comput 64(7):2036–2048
Khosravi F, Reimann F, Glaß M, Teich J (2014) Multi-objective local-search optimization using reliability importance measuring. In: Proceedings of the 51st annual design, automation conference, pp 1–6
Koren I, Krishna CM (2007) Fault-tolerant systems. Morgan Kaufmann, San Francisco
Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Res Logist Q 2:83–97
Li L, Degalahal V, Vijaykrishnan N, Kandemir M, Irwin M (2004) Soft error and energy consumption interactions: a data cache perspective. In: Proceedings-international symposium on low power electronics and design, pp 132–137
Lifa A, Eles P, Peng Z, Izosimov V (2010) Hardware/software optimization of error detection implementation for real-time embedded systems. In: Proceedings of the international conference on hardware/software codesign and system synthesis, pp 41–50
Lyons R, Vanderkulk W (1962) The use of triple-modular redundancy to improve computer reliability. IBM J Res Dev 6(2):200–209
Nikolov D, Larsson E (2016) Optimizing the level of confidence for multiple Jobs. IEEE Trans Comput 65(4):1239–1252
Pop P, Izosimov V, Eles P, Peng Z (2009) Design optimization of time- and cost-constrained fault-tolerant embedded systems with checkpointing and replication. IEEE Trans Very Large Scale Integr Syst 17(3):389–402
Pradhan DK (1996) Fault-tolerant computer system design. Prentice-Hall Inc., Upper Saddle River
Pradhan D, Vaidya N (1994) Roll-forward checkpointing scheme: a novel fault-tolerant architecture. IEEE Trans Comput 43(10):1163–1174
Raghunathan B, Turakhia Y, Garg S, Marculescu D (2013) Cherry-picking: exploiting process variations in dark-silicon homogeneous chip multi-processors. In: Proceedings of the design, automation test on European conference exhibition, pp 39–44
Reimann F, Glaß M, Lukasiewycz M, Keinert J, Haubelt C, Teich J (2008) Symbolic voter placement for dependability-aware system synthesis. In: Proceedings of the international conference on hardware/software codesign and system synthesis, pp 237–242
Rubio-Largo Á, Vega-Rodríguez MA, González-Álvarez DL (2016) A hybrid multiobjective memetic metaheuristic for multiple sequence alignment. IEEE Trans Evol Comput 20(4):499–514
Salehi M, Ejlali A, Al-Hashimi BM (2016a) Two-phase low-energy N-modular redundancy for hard real-time multi-core systems. IEEE Trans Parallel Distrib Syst 27(5):1497–1510
Salehi M, Tavana MK, Rehman S, Shafique M (2016b) Two-state checkpointing for energy-efficient fault tolerance in hard real-time system. IEEE Trans Very Large Scale Integr Syst 24(7):2426–2437
Shen X, Yao X (2015) Mathematical modelling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inf Sci 298:198–224
Stralen P, Pimentel A (2012) A SAFE approach towards early design space exploration of fault-tolerant multimedia MPSoCs. In: Proceedings of the international conference on hardware/software codesign and system synthesis, pp 393–402
Tersi L, Fantozzi S, Stagni R (2015) Characterization of the performance of memetic algorithms for the automation of bone tracking with fluoroscopy. IEEE Trans Evol Comput 19(1):19–30
Vadlamani R, Zhao J, Burleson W, Tessier R (2010) Multicore soft error rate stabilization using adaptive dual modular redundancy. In Proceedings of the design, automation test in European conference exhibition, 2010, pp 27–32
Wang P, Emmerich M, Li R, Tang K, Baeck T, Yao X (2015) Convex hull-based multi-objective genetic programming for maximizing receiver operating characteristic performance. IEEE Trans Evol Comput 19(2):188–200
Wang Z, Tang K, Yao X (2010) A memetic algorithm for multi-level redundancy allocation. IEEE Trans Reliab 5(4):754–765
Yuan Y, Xu H (2015) Multiobjective flexible job shop scheduling using memetic algorithms. IEEE Trans Autom Sci Eng 12(1):336–353
Yuan B, Li B, Weise T, Yao X (2014) A new memetic algorithm with fitness approximation for the defect-tolerant logic mapping in crossbar-based nanoarchitectures. IEEE Trans Evol Comput 18(6):846–859
Yuan B, Li B, Chen H, Yao X (2016) Defect-and variation-tolerant logic mapping in nanocrossbar using bipartite matching and memetic algorithm. IEEE Trans Very Large Scale Integr Syst 24(9):2813–2826
Zhong F, Yuan B, Li B (2016) A hybrid evolutionary algorithm for multiobjective variation tolerant logic mapping on nanoscale crossbar architectures. Appl Soft Comput 38:955–966
Ziv A, Bruck J (1997) Performance optimization of checkpointing schemes with task duplication. IEEE Trans Comput 46(12):1381–1386